Bayesian Abductive Logic Programs

Sindhu Raghavan and **Raymond Mooney**
Department of Computer Science
The University of Texas at Austin
1 University Station C0500
Austin, TX 78712-0233, USA
{sindhu,mooney}@cs.utexas.edu

Abstract

In this paper, we introduce Bayesian Abductive Logic Programs (BALPs), a new formalism that integrates Bayesian Logic Programs (BLPs) and Abductive Logic Programming (ALP) for abductive reasoning. Like BLPs, BALPs also combine first-order logic and Bayesian networks. However, unlike BLPs that use logical deduction to construct Bayes nets, BALPs employ logical abduction. As a result, BALPs are more suited for solving problems like plan/activity recognition and diagnosis that require abductive reasoning. First, we present the necessary enhancements to BLPs in order to support logical abduction. Next, we apply BALPs to the task of plan recognition and demonstrate its efficacy on two data sets. We also compare the performance of BALPs with several existing approaches for abduction.

1 Introduction

Abduction is defined as the process of finding the best explanation for a set of observations (Peirce 1958). It is widely used in tasks such as activity recognition, plan recognition, and diagnosis, that require inferring cause from effect. Most previous approaches to abductive reasoning have been based on first-order logic and determine a small set of assumptions sufficient to deduce the observations (Pople 1973; Levesque 1989). Abductive Logic Programming (ALP) (Kakas, Kowalski, and Toni 1993) is one of the popular frameworks for performing abductive reasoning in first-order logic where the background theory is restricted to Horn clauses. While these logic-based approaches handle structured representations, they are unable to handle uncertainty in the observations or background knowledge and are incapable of estimating the likelihood of alternative explanations. Another popular approach to abduction involves using Bayesian networks to compute the posterior probability of alternative explanations given the observations (Pearl 1988). A major limitation of this approach is that it cannot handle structured representations involving relations amongst multiple entities since Bayes nets are essentially propositional in nature.

Recently, there has been a proliferation of new formalisms for integrating first-order logic and probabilistic graphical models to develop statistical relational models for knowledge representation, inference, and learning (Getoor and Taskar 2007). Of these formalisms, Markov Logic Networks (MLNs) (Richardson and Domingos 2006), which combine first-order logic and undirected graphical models (Markov nets) have been used for abductive plan recognition by Kate and Mooney (2009). Since MLNs employ deduction as the logical inference mechanism, Kate and Mooney (2009) proposed an approach to adapt MLNs for abductive reasoning. Like MLNs, most SRL formalisms use deduction for logical inference, and hence cannot be used effectively for reasoning requiring logical abduction.

In this paper, we propose Bayesian Abductive Logic Programs (BALPs), a new formalism that integrates Bayesian Logic Programs (BLPs) (Kersting and De Raedt 2001; 2007) and Abductive Logic Programming. Like most SRL formalisms, BLPs also use deduction for logical inference, and hence cannot be used effectively for abductive reasoning. As a result, we enhance BLPs to support abductive reasoning by employing logical abduction instead of deduction. Like BLPs, BALPs combine first-order logic and directed graphical models (Bayesian networks). Like all SRL formalisms, BALPs also integrate the strengths of both first-order logic and probabilistic graphical models, thus overcoming the limitations of the approaches mentioned above.

First, we briefly review abduction and BLPs. Next, we describe our enhancements to BLPs in order to support abductive reasoning. We then apply BALPs to the task of plan recognition and demonstrate its efficacy on two data sets. Finally, we present experimental comparison of BALPs with several existing approaches for abduction and discuss the results.

2 Background

This section briefly reviews relevant prior research on abduction and BLPs.

2.1 Abduction

Plan recognition, diagnosis, language interpretation and many other tasks can be viewed as types of abductive reasoning (Charniak and McDermott 1985). In a logical framework, abduction, is usually defined as follows (Pople 1973):

- **Given:** Background knowledge B and observations O, both represented as sets of formulae in first-order logic,

where O is typically restricted to a conjunction of ground literals.

- **Find:** A hypothesis H, also a set of logical formulae, such that $B \cup H \not\models \bot$ and $B \cup H \models O$.

Here $\models$ means logical entailment and $\bot$ means false, i.e. find a set of assumptions that is consistent with the background theory and explains the observations. There are generally many hypotheses H that explain a particular set of observations O. The best hypothesis is typically selected based on the size (simplicity) of H, following Occam's Razor. Often the background knowledge B is restricted to a set of Horn clauses and the hypothesis H is restricted to a set of ground atoms, resulting in abductive logic programming (Kakas, Kowalski, and Toni 1993). Several researchers have applied logical abduction to tasks like plan recognition and diagnosis, such as Ng and Mooney (1992). The primary problem with the logical approach is that it does not handle uncertainty.

Another popular approach to abduction employs Bayesian networks (Pearl 1988), in which the background knowledge and its uncertainty is encoded in a parameterized directed graph. Then, given a set of observations, probabilistic inference is used to compute the posterior probability of alternative explanations. However, a major limitation of Bayesian networks is that they are essentially propositional and cannot handle structured representations.

Recently, Kate and Mooney (2009) proposed an approach to adapt MLNs for abduction. Their approach requires adding reverse implications for every rule in the knowledge base to support logical abduction. Further, to support "explaining away", they add mutual exclusivity constraints on the transformed rules. Explaining away refers to the phenomenon that evidence for one explanation decreases confidence in alternative competing explanations (Pearl 1988). However, the addition of these rules increases the size and complexity of MLNs, resulting in a computationally intensive problem. Furthermore, adding reverse implications to the MLN frequently results in rules with multiple existential variables, causing a combinatorial explosion in the construction of the ground Markov network.

2.2 Bayesian Logic Programs

Bayesian logic programs (BLPs) (Kersting and De Raedt 2001; 2007) can be considered as templates for constructing *directed* graphical models (Bayes nets). Given a knowledge base as a special kind of logic program, standard logical inference (SLD resolution) is used to automatically construct a Bayes net for a given problem. More specifically, given a set of facts and a query, all possible Horn-clause proofs of the query are constructed and used to build a Bayes net for answering a specific query. Standard probabilistic inference techniques are then used to compute the most probable answer.

More formally, a BLP consists of a set of *Bayesian clauses*, definite clauses of the form $A|A_1, A_2, A_3, ...A_n$, where $n \geq 0$ and A, A_1, A_2, A_3,.....,A_n are *Bayesian predicates* (defined below). A is called the head of the clause (head(c)) and (A_1, A_2, A_3,....,A_n) is the body (body(c)). When $n = 0$, a Bayesian clause is a fact. Each Bayesian clause c is assumed to be universally quantified and range restricted, i.e $variables\{head\} \subseteq variables\{body\}$, and has an associated *conditional probability distribution* cpd(c) = P(head(c)|body(c)).

A *Bayesian predicate* is a predicate with a finite domain, and each ground literal for a Bayesian predicate represents a random variable. Associated with each Bayesian predicate is a combining rule such as *noisy-or* or *noisy-and* that maps a finite set of cpds into a single cpd (Pearl 1988). Let A be a Bayesian predicate defined by two Bayesian clauses, $A|A_1, A_2, A_3, ...A_n$ and $A|B_1, B_2, B_3, ...B_n$, where cpd_1 and cpd_2 are their cpd's. Let θ be a substitution that satisfies both clauses. Then, in the constructed Bayes net, directed edges are added from the nodes for each $A_i\theta$ and $B_i\theta$ to the node for $A\theta$. The combining rule for A is used to construct a single cpd for $A\theta$ from cpd_1 and cpd_2. The probability of a joint assignment of truth values to the final set of ground propositions is then defined in the standard way for a Bayes net: P(X) = $\prod_i P(X_i|Pa(X_i))$, where $X = X_1, X_2, ..., X_n$ represents the set of random variables in the network and $Pa(X_i)$ represents the parents of X_i. The cpds for Bayesian clauses can be learned using the methods described by Kersting and De Raedt (2007). Once a ground network is constructed, standard probabilistic inference methods can be used to answer various types of queries (Koller and Friedman 2009).

3 Bayesian Abductive Logic Programs

We now describe our approach to adapt the BLP framework to abductive reasoning. In the BLP framework, a Bayesian clause specifies a *directed* relationship between its body and head rather than simply a deductive one. The clauses in the knowledge base are then used to specify the structure of a Bayesian network and normal probabilistic inference can be used to compute explanations based on the evidence. However, BLPs use deductive inference to construct the ground Bayes net. This is insufficient for logically abductive tasks like plan recognition since the known facts are insufficient to support the derivation of deductive proof trees for the necessary queries. Therefore, logical *ab*duction must be used to construct the proof trees needed to determine the structure of the ground network. By allowing literals to be assumed in order to complete a proof, alternative explanations for a query can be derived. Therefore, we derive a complete set of abductive proof trees for a query using the method originally proposed by Stickel (1988).

Let O_1, O_2, ..., O_n be the set of observations. We recursively compute abductive proofs for each observation literal by backchaining on each O_i until every literal in the proof is proven or assumed. A query literal is said to be assumed if it cannot unify with the head of any rule in the knowledge base. A literal is said to be proven if it unifies with the head of some rule in the knowledge base. Since multiple plans/actions could generate the same observation, an observation literal could unify with the head of multiple rules in the knowledge base. For such a literal, we compute alternate abductive proofs. The resulting abductive proof trees

are then used to build the structure of the ground Bayes net using the standard approach for BLPs.

We now illustrate the process of computing abductive proofs with an example. Consider the following Bayesian logic program:

inst(G,going) | *inst(B,shopping)* , *go-step(B,G)*
goer(G,P) | *inst(S,shopping)* , *go-step(S,G)* , *shopper(S,P)*

and the set of ground literals representing the observations:

inst(go1,going), *goer(go1,john1)*

For each observation literal, we recursively backchain to generate abductive proof trees. For example, when we backchain on the literal inst(go1,going), we obtain the subgoals inst(B,shopping) and go-step(B,go1). These subgoals become assumptions since no observations or heads of clauses unify with them. Since B is an existentially quantified variable, we replace it with a Skolem constant a1 to obtain the ground assumptions inst(a1,shopping) and go-step(a1,go1). Similarly, when backchaining on goer(go1,john1), we generate the subgoals inst(S,shopping), go-step(S,go1), shopper(S,john1). Matching the first two to existing assumptions, we ground these assumptions to inst(a1,shopping), go-step(a1,go1), shopper(a1,john1). After generating all abductive proofs for all observation literals, we construct a Bayesian network. Figure 1 gives the Bayesian network constructed for this simple example.

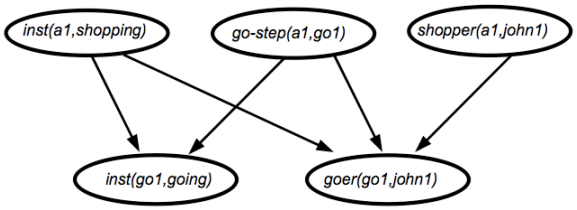

Figure 1: Bayesian network constructed for the given example.

We used noisy-and and noisy-or models to specify the cpds in the ground Bayesian network. Noisy-and is used to specify the cpd for combining evidence from the conjuncts in the body of a clause. It is also possible to explicitly specify the entries in the cpd for each clause or learn these parameters from the data. However, as the number of literals in the body of the clause increases, the number of entries in the cpd increases exponentially. It might not always be feasible to specify or even learn these entries from the data. Hence, we used the noisy-and model as it compactly encodes the cpd with fewer parameters. We used the noisy-or model to specify the cpd for combining the disjunctive contributions from different ground clauses with the same head. Noisy-or is a standard approach for encoding a cpd to support "explaining away". Currently, we set the weights manually using a simple heuristic; however, given sufficient training data, they could be learned using the methods described by Kersting and De Raedt (2007).

Given the constructed Bayes net and a set of observations, we compute the best explanation using standard methods for computing the Most Probable Explanation (MPE) (Pearl 1988), which assigns values to the unobserved nodes in the network that maximize the posterior probability of the joint assignment given the observations. In some domains, it may be useful to compute several alternative explanations. To support this, we compute the k-MPE (Nilsson 1998), which constructs the top k explanations for a Bayesian network given the set of observations. We used Elvira, a Java based environment for probabilistic graphical models (Elvira-Consortium 2002) to perform probabilistic reasoning on Bayesian networks.

4 Experimental Evaluation

This section presents experiments that compared BALPs with existing approaches on two plan-recognition data sets.

4.1 Datasets

Dataset 1: We used a dataset for plan recognition previously used to evaluate abductive systems [1] (Ng and Mooney 1992; Charniak and Goldman 1991). In this task, character's higher-level plans must be inferred in order to explain their observed actions described in a narrative text. A logical representation of the literal meaning of the narrative text is given for each example. An example of a narrative text is: "Bill went to the liquor-store. He pointed a gun at the owner." The dataset consists of 25 development and 25 test examples. The development data was constructed by Goldman (1990) and the testing data was later added by Ng and Mooney (1992). Their logical representations which form the observations to be explained contain an average of 12.6 literals per example.

The background knowledge-base was initially constructed for the ACCEL system (Ng and Mooney 1991) to work with the 25 development examples. It was constructed such that the high-level plans (like shopping and robbing) together with appropriate role-fillers (such as someone being the shopper of a shopping plan or a robber of a robbing plan) imply the input literals representing the observed actions (like going to a store and pointing a gun). The plans in the knowledge base include shopping, robbing, restaurant dining, traveling in a vehicle (bus, taxi or plane), partying and jogging. Some narratives involve more than a single plan.

Dataset 2: We also used the Monroe dataset, an artificially generated plan-recognition dataset in the emergency response domain by Blaylock and Allen (2005a). This domain includes top level plans like setting up a temporary shelter, clearing a road wreck, and providing medical attention to victims. The plan-recognition task involves inferring a single top level plan based on a set of observed actions. While the original dataset had 5,000 examples, we chose to use the first 1,000 examples in our evaluation. Each example instantiates one of 10 top-level general plans and consists of a set of ground literals describing its execution. There are an average of 10.19 observation literals per example. We also

[1] This data can be downloaded from `http://www.cs.utexas.edu/~ml/accel.html`

used a separate development set of 300 examples for tuning the parameters of the different systems.

While we were able to experimentally evaluate the performance of BALPs on the Monroe data and compare some of our results to those of Blaylock and Allen (2005b), we were unable to compare to the performance of Kate and Mooney's (2009) MLN approach on this data due to implementation issues. Their approach resulted in an MLN with several rules containing multiple existentially quantified variables. These rules resulted in an exponential number of possible groundings, eventually leading to a memory overflow. In order to compare the performance of BALPs with this previous MLN approach, we slightly modified the original Monroe domain. We then used the system developed by Blaylock and Allen (2005a) to generate another plan corpus. The resulting dataset had 1,000 examples, with an average of 10.56 observations per example. We again created a separate development set of 300 examples for tuning the parameters. We refer to this as the "modified-Monroe" dataset.

The plan generation system uses the SHOP2 planner (Nau et al. 2003). Since SHOP2 is a hierarchical transition network (HTN) planner, the emergency response domain is represented in the HTN format. We constructed a logical knowledge base representing the domain knowledge encoded in the HTN.

4.2 Methodology

For dataset 1, we compared the performance of BALPs with that of the MLN approach by Kate and Mooney (2009) and ACCEL (Ng and Mooney 1992), a purely logic-based system. ACCEL uses a metric to guide its search for selecting the best explanation. For the plan recognition task, it can use two different metrics. The first is *simplicity*, which selects the explanation of the smallest size, i.e. the one with the fewest number of assumptions. The second is *coherence*, which selects the explanation that maximally connects the input observations. This second metric is specifically geared towards the task of text interpretation to measure *explanatory coherence* (Ng and Mooney 1990), i.e., how well the input sentences are tied together in the final interpretation. Currently, this bias has not been incorporated in either the BALP or MLN approach. For dataset 2, we compared the performance of BALPs with that of the system developed by Blaylock and Allen (2005b) on the Monroe dataset. We also compared the performance of BALPs with that of the MLN approach on the modified-Monroe dataset.

For abductive MLNs, we used the method described by Kate and Mooney (2009) to automatically add clauses to the background knowledge base for dataset2 (modified-Monroe). We used Alchemy, an open-source MLN software package to perform probabilistic inference and learning.[2] Alchemy requires specifying types for the arguments of each predicate, these are used along with provided constants of each type to ground the MLN into a Markov network. We also specialized some predicates in the original knowledge bases to improve the efficiency of the resulting MLN. This extra knowledge engineering was required to make MLN inference tractable. We would like to note that even though the original domain for dataset 2 was modified to make it work with MLNs, there were still several reverse implications in the MLN with multiple existentially quantified variables, which made the inference intractable. As a result, we had to remove these reverse implications from the MLN. However, we note that these rules are not essential for inferring the high level plans. For dataset 1, we used the MLN developed by Kate and Mooney (2009) without any modification. For BALPs, we were able to use the original knowledge bases constructed for the two datasets, and used the method described in section 3 to perform plan recognition.

For the MLN-based system, we were unable to effectively learn clause weights. For dataset 1, we found that the 25 development examples were too few to learn useful weights. Hence, we used the manually-tuned weights of Kate and Mooney (2009) for dataset 1. For dataset 2 (modified-Monroe), it was intractable to run Alchemy's existing weight-learners, as the domain was quite large. Hence, we manually set the weights based on preliminary experiments on the development set. We put soft weights (1, 2 or 3) on the reverse implication clauses and hard weights for all the mutual exclusivity clauses. In addition, we put small negative weights (-1) on unit clauses for all predicates to encode the prior probability that most facts are false *a priori*. For BALPs, we hand-tuned the parameters for the cpds on the development set for both datasets. For the noisy-and parameters, we treated all literals in the body of the clause as having the same effect on the head, and set equal weights (around .9). For the noisy-or parameters, we treated different plans resulting in the same observation as equally likely and set equal weights (around .9). However, to disambiguate between conflicting plans, we set different priors for high level plans. For both BALPs and abductive MLNs, we tried to manually tune the weights/parameters to maximize performance on the development set.

In dataset 1, the plan-recognition task involved computing the set of literals that best explained the observations. We used MPE inference (Pearl 1988) to compute the best explanation for both BALPs and MLNs. However, for dataset 2, the task involved inferring a *single* top level plan that best explained the observations. Hence, we computed the marginal probabilities for the plan predicates and picked the single plan with the highest marginal probability as the best explanation for the observations.

The observation set for dataset 2 includes all actions executed to achieve the top level plan. In order to evaluate performance for partially observed plans, we also performed plan recognition given only subsets of the final actions. Specifically, we report results after observing the first 25%, 50%, 75%, and 100% of the executed actions. The observed literals for dataset 1 are already incomplete and were used as is.

We now describe the evaluation metrics. For dataset 1, we compared the plans inferred by different systems with the ground truth to compute *precision* and *recall* scores. Precision measures the fraction of the predicted plans that are present in the ground truth, while recall measures the frac-

[2] http://alchemy.cs.washington.edu

	ACCEL-Coh	ACCEL-Sim	BALP	MLN
Precision	89.39	66.45	72.07	67.31
Recall	89.39	52.32	85.57	68.10
F-measure	89.39	58.54	78.24	67.70

Table 1: Results for the different abductive systems on dataset 1."ACCEL-Coh" refers to the ACCEL with coherence metric and "ACCEL-Sim" refers to ACCEL with simplicity metric.

tion of the plans in the ground truth that are predicted. We also computed the *F-measure*, the harmonic mean of precision and recall. For dataset 2, since both the inferred solution and the ground truth have a single plan, we simply compared the inferred plan with the correct plan to compute an accuracy score. When computing precision, recall, and accuracy, partial credit was given for predicting the correct plan predicate with only a subset of its correct arguments. A point was rewarded for inferring the correct plan predicate, then, given the correct predicate, an additional point was rewarded for each correct argument. For example, if the correct plan was $plan_1(a_1, a_2)$ and the inferred plan was $plan_1(a_1, a_3)$, the accuracy score was 66.67%.

For dataset 1, Kate and Mooney (2009) compared the inferred literals with those in the ground truth to compute precision and recall scores. However, the explanations generated by the MLN include additional facts implied by the minimal explanation. To fairly compare the different systems, we constructed high level plans from the predicted and true ground literals and then computed plan-level precision and recall scores. As a result, the performance scores reported for dataset 1 are not comparable to those reported by Kate and Mooney (2009).

The experimental methodology employed by Blaylock and Allen (2005b) was somewhat different, so we could not compare our results on the Monroe data directly with theirs. However, we were able to compare to their results using their *convergence* score for plan schema recognition. In their paper, the plan schema refers to the top level plan predicate without considering its arguments, and convergence score refers to the fraction of examples for which the plan schema was correctly predicted when given *all* of the observations.

4.3 Results and Discussion

Table 1 shows the results for dataset 1. We found that BALPs performed better than both ACCEL-Simplicity and the MLN based system; however, ACCEL-Coherence out-performed BALPs and the other systems. Since the coherence metric incorporates extra criteria specific to interpreting narrative text, this bias would need to be included in the probabilistic models to make them more competitive with ACCEL-Coherence. Further, the coherence metric is specific to narrative interpretation and not applicable to plan recognition in general.

Table 2 shows the results for BALPs and the MLN based system on dataset 2 (modified-Monroe). BALPs consistently outperform the MLN-based system on this dataset. The difference in performance is small when there are fewer

	Acc-100	Acc-75	Acc-50	Acc-25
BALP	91.80	56.70	25.25	9.25
MLN	79.13	36.83	17.46	6.91

Table 2: Results for BALP and MLN based systems on dataset 2 (modified-Monroe). Acc-i is the acurracy when given the first $i\%$ of the observations.

	BALPs	Blaylock and Allen
Convergence	98.80	94.2

Table 3: Convergence score for BALPs and the system developed by Blaylock and Allen on dataset 2 (Monroe).

observations (25%); however, with 100% observations, the performance of BALPs is significantly better than that of abductive MLNs.

Table 3 shows the convergence score for BALPs and the system developed by Blaylock and Allen (2005b) for instantiated goal/plan recognition. BALPs outperform Blaylock and Allen's system on this metric. Note that while the convergence score for BALPs is averaged across 1,000 examples, the convergence score reported by Blaylock and Allen is averaged across 500 examples.

Overall, BALPs outperformed most existing approaches on these datasets. Unlike the MLN based approach by Kate and Mooney (2009), it is possible to use an existing knowledge base without any modification to perform abductive reasoning in BALPs. Unlike the statistical approach by Blaylock and Allen (2005b), BALPs are capable of jointly predicting the plan and its arguments simultaneously. Even though Blaylock and Allen perform instantiated plan recognition, it is actually done in two separate steps. The first step predicts the plan schema and the second step predicts the arguments given the schema. We believe that BALP's ability to perform joint instantiated goal/plan prediction has led to its superior performance on the Monroe dataset. In all, we found BALPs to be a promising approach for abductive reasoning.

5 Related Work

Charniak and Goldman (1989; 1991) also developed an approach to automatically construct Bayesian networks for plan recognition. Their work is similar to BALPs, but special purpose procedures were used to construct the necessary ground networks rather than using a general-purpose probabilistic predicate logic like MLNs and BLP/BALPs. As discussed above, Blaylock and Allen (2003; 2004; 2005b) train special-purpose n-gram models to perform separate plan-schema and plan-argument prediction, compared to the joint prediction supported by a probabilistic predicate logic. Abductive reasoning has also been applied to the problem of modeling inhibition in metabolic networks (Tamaddoni-Nezhad et al. 2006; Chen, Muggleton, and Santos 2008). Further, Chen et. al (2008) extend stochastic logic programs (Muggleton 2003) to incorporate abductive reasoning. Like BALPs, abductive SLPs are based on possible world semantics. On the other hand, Sato (1995) has developed a pro-

gramming language based on distribution semantics called PRISM, which is capable of performing abductive reasoning. It would be interesting to compare the performance of these approaches with that of BALPs on the plan recognition task.

6 Future Work and Conclusions

This paper has introduced BALPs, a new SRL formalism that integrates Bayesian Logic Programs and Abductive Logic Programming. Empirical evaluations on two different plan recognition datasets demonstrated that BALPs outperform most existing approaches on these domains.

In the future, we plan to explore learning the BALP parameters (i.e. cpds) automatically from training data. Since BALPs currently only handle discrete-valued domains, we would also like to extend them to handle continuous variables. We are also exploring alternative approaches to performing logical abductive reasoning using MLNs in addition to Kate and Mooney's (2009) approach employed in the results presented here. As shown by Richardson and Domingos (2006), any BLP can be modeled by a semantically equivalent MLN. Given our success with BALPs for plan recognition, we are developing an alternative MLN formulation that more closely models the BALP approach. Preliminary results on this alternative MLN formulation for the modified Monroe data are competitive with our current BALP results.

Acknowledgments

This research was funded by MURI ARO grant W911NF-08-1-0242. Experiments were run on the Mastodon Cluster, provided by NSF Grant EIA-0303609.

References

Blaylock, N., and Allen, J. 2003. Corpus-based, statistical goal recognition. In *IJCAI-03*, 1303–1308.

Blaylock, N., and Allen, J. F. 2004. Statistical goal parameter recognition. In *ICAPS-04*, 297–305.

Blaylock, N., and Allen, J. 2005a. Generating artificial corpora for plan recognition. In *UM-05*. Springer.

Blaylock, N., and Allen, J. 2005b. Recognizing instantiated goals using statistical methods. In *in: G. Kaminka (Ed.), Workshop on MOO-05*, 79–86.

Charniak, E., and Goldman, R. P. 1989. A semantics for probabilistic quantifier-free first-order languages, with particular application to story understanding. In *IJCAI-89*.

Charniak, E., and Goldman, R. 1991. A probabilistic model of plan recognition. In *AAAI-91*, 160–165.

Charniak, E., and McDermott, D. 1985. *Introduction to Artificial Intelligence*. Reading, MA: ADDISON.

Chen, J.; Muggleton, S.; and Santos, J. 2008. Learning probabilistic logic models from probabilistic examples. *Machine Learning* 73(1):55–85.

Elvira-Consortium. 2002. Elvira: An environment for probabilistic graphical models. In *Proceedings of the Workshop on Probabilistic Graphical Models*.

Getoor, L., and Taskar, B., eds. 2007. *Introduction to Statistical Relational Learning*. Cambridge, MA: MITP.

Goldman, R. P. 1990. *A Probabilistic Approach to Language Understanding*. Ph.D. Dissertation, Dept. of Computer Science, Brown University, RI. Technical Report CS-90-34.

Kakas, A. C.; Kowalski, R. A.; and Toni, F. 1993. Abductive logic programming. *Journal of Logic and Computation* 2(6):719–770.

Kate, R. J., and Mooney, R. J. 2009. Probabilistic abduction using Markov logic networks. In *IJCAI-09 Workshop on Plan, Activity, and Intent Recognition*.

Kersting, K., and De Raedt, L. 2001. Towards combining inductive logic programming with Bayesian networks. In *ILP-01*, 118–131.

Kersting, K., and De Raedt, L. 2007. Bayesian logic programming: Theory and tool. In Getoor, L., and Taskar, B., eds., *An Introduction to Statistical Relational Learning*. Cambridge, MA: MITP.

Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Levesque, H. J. 1989. A knowledge-level account of abduction. In *IJCAI-89*, 1061–1067.

Muggleton, S. 2003. Learning structure and parameters of stochastic logic programs. In *ILP-02*, 198–206.

Nau, D.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. Shop2: An HTN planning system. *Journal of Artificial Intelligence Research* 20:379–404.

Ng, H. T., and Mooney, R. J. 1990. The role of coherence in abductive explanation. In *AAAI-90*, 337–442.

Ng, H. T., and Mooney, R. J. 1991. An efficient first-order Horn-clause abduction system based on the ATMS. In *AAAI-91*, 494–499.

Ng, H. T., and Mooney, R. J. 1992. Abductive plan recognition and diagnosis: A comprehensive empirical evaluation. In *KR-92*, 499–508.

Nilsson, D. 1998. An efficient algorithm for finding the M most probable configurations in probabilistic expert systems. *Statistics and Computing* 8:159–173.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. CA: MKP.

Peirce, C. S. 1958. *Collected Papers of Charles Sanders Peirce*. Cambridge, Mass.: MITP.

Pople, H. E. 1973. On the mechanization of abductive logic. In *IJCAI-73*, 147–152.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62:107–136.

Sato, T. 1995. A statistical learning method for logic programs with distribution semantics. In *ICLP-95*, 715–729. MIT Press.

Stickel, M. E. 1988. A Prolog-like inference system for computing minimum-cost abductive explanations in natural-language interpretation. Technical Report Tech. Note 451, SRI International, CA.

Tamaddoni-Nezhad, A.; Chaleil, R.; Kakas, A.; and Muggleton, S. 2006. Application of abductive ILP to learning metabolic network inhibition from temporal data. *Machine Learning* 64(1-3):209–230.